

**DESSERT  
FINANCE**



**DessertSwap (DSRT)**

**AI LIGHT AUDIT**

Performed at block **SAMPLE**

PERFORMED BY DESSERT FINANCE

FOR CONTRACT ADDRESS: `0x6D6959ba76e259C4fd50f8b638b07ce17E1c6312`

## INITIAL DISCLAIMER

Dessert Finance provides due-diligence project audits for various projects. Dessert Finance in no way guarantees that a project will not remove liquidity, sell off team supply, or otherwise exit scam.

Dessert Finance does the legwork and provides public information about the project in an easy-to-understand format for the common person.

Agreeing to an audit in no way guarantees that a team will not remove *all* liquidity (“Rug Pull”), remove liquidity slowly, sell off tokens, quit the project, or completely exit scam. There is also no way to prevent private sale holders from selling off their tokens. It is ultimately your responsibility to read through all documentation, social media posts, and contract code of each individual project to draw your own conclusions and set your own risk tolerance.

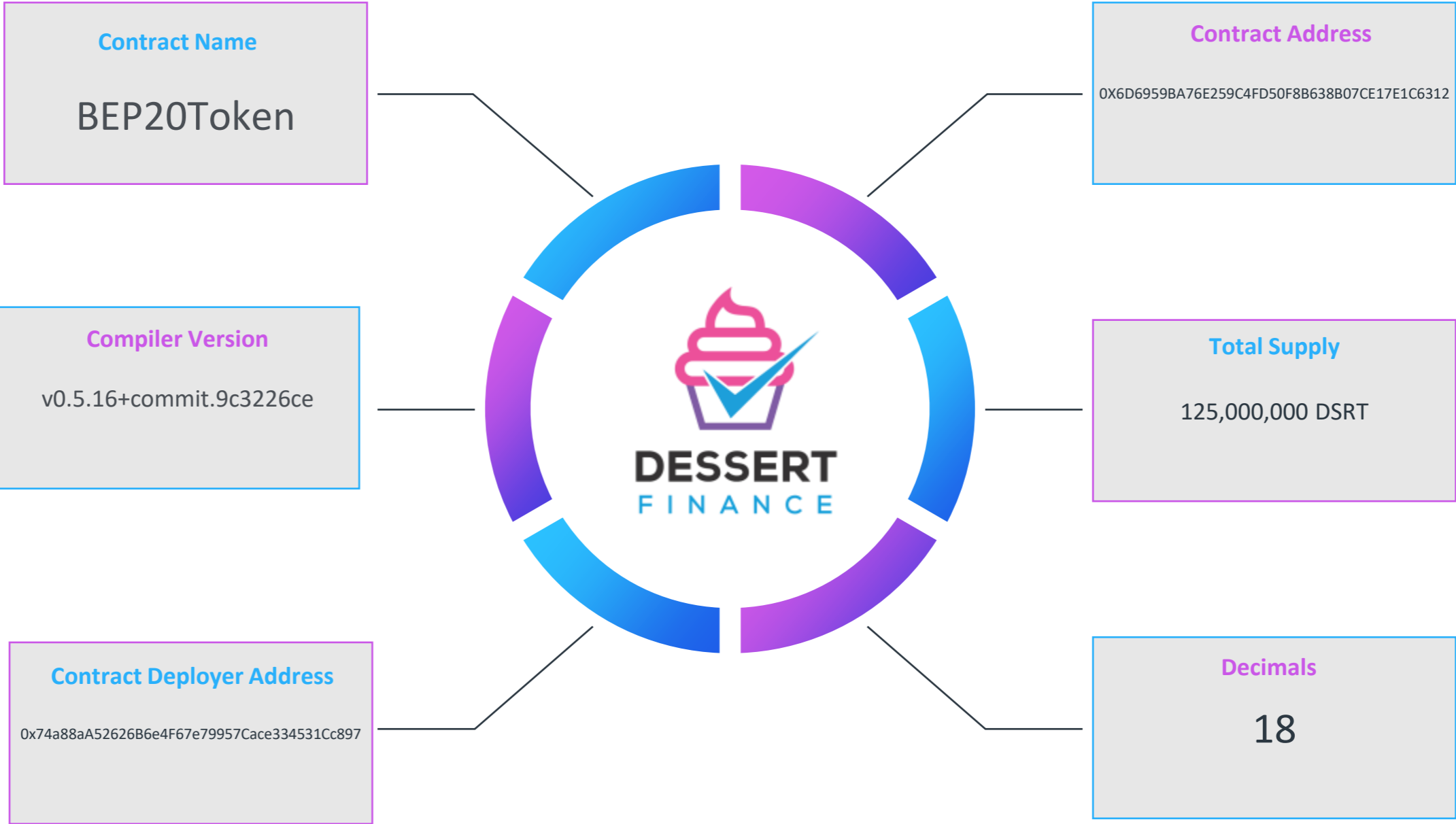
Dessert Finance in no way takes responsibility for any losses, nor does Dessert Finance encourage any speculative investments. The information provided in this audit is for information purposes only and should not be considered investment advice. Dessert Finance does not endorse, recommend, support, or suggest any projects that have been audited. An audit is an informational report based on our findings, We recommend you do your own research, we will never endorse any project to invest in.

# Table of Contents



1. Contract Code Audit – Token Overview
2. BEP-20 Contract Code Audit – Overview
3. BEP-20 Contract Code Audit – Vulnerabilities Checked
4. Disclaimers

# Contract Code Audit – Token Overview



# BEP-20 Contract Code Audit – Overview

Dessert Finance was commissioned to perform an audit on DessertSwap (DSRT)

```
Submitted for verification at BscScan.com on 2021-02-23  
pragma solidity 0.5.16;  
  
interface IBEP20 {  
    /**  
     * @dev Returns the amount of tokens in existence.  
     */  
    function totalSupply() external view returns (uint256);  
  
    /**  
     * @dev Returns the token decimals.  
     */  
    function decimals() external view returns (uint8);  
  
    /**  
     * @dev Returns the token symbol.  
     */  
    function symbol() external view returns (string memory);  
  
    /**  
     * @dev Returns the token name.  
     */  
    function name() external view returns (string memory);  
  
    /**  
     * @dev Returns the bag token owner.  
     */  
    function getOwner() external view returns (address);  
  
    /**  
     * @dev Returns the amount of tokens owned by 'account'.  
     */  
    function balanceOf(address account) external view returns (uint256);  
  
    /**  
     * @dev Moves 'amount' tokens from the caller's account to 'recipient'.  
     * Returns a boolean value indicating whether the operation succeeded.  
     * Emits a [Transfer] event.  
     */  
    function transfer(address recipient, uint256 amount) external returns (bool);  
  
    /**  
     * @dev Returns the remaining number of tokens that 'spender' will be  
     * allowed to spend on behalf of 'owner' through [transferFrom]. This is  
     * zero by default.  
     * This value changes when [approve] or [transferFrom] are called.  
     */  
    function allowance(address owner, address spender) external view returns (uint256);  
  
    /**  
     * @dev Sets 'amount' as the allowance of 'spender' over the caller's tokens.  
     * Returns a boolean value indicating whether the operation succeeded.  
     * IMPORTANT: Beware that changing an allowance with this method brings the risk  
     * that someone may use both the old and the new allowance by unfortunate  
     * transaction ordering. One possible solution to mitigate this race  
     * condition is to first reduce the spender's allowance to 0 and set the  
     * desired value afterwards.  
     * See https://github.com/ethereum/EIPs/issues/20#issuecomment-263520729  
     */  
    function approve(address spender, uint256 amount) external returns (bool);  
  
    /**  
     * @dev Emits an [Approval] event.  
     */  
}
```

## Contract Address

0x6D6959ba76e259C4fd50f8b638b07ce17E1c6312

## TokenTracker

DessertSwap (DSRT)

## Contract Creator

0x74a88aa52626b6e4f67e79957cace334531cc897

## Source Code

Contract Source Code Verified

## Contract Name

BEP20Token

## Other Settings

default evmVersion, None

## Compiler Version

v0.5.16+commit.9c3226ce

## Optimization Enabled

Yes with 200 runs

Code is truncated to fit the constraints of this document.

[The code in its entirety can be viewed here.](#)

# BEP-20 Contract Code Audit – Vulnerabilities Checked

Vulnerability Tested	Scan	Result
Compiler Errors	Complete	✓ Low / No Risk
Outdated Compiler Version	Complete	✓ Low / No Risk
Integer Overflow	Complete	✓ Low / No Risk
Integer Underflow	Complete	✓ Low / No Risk
Floating Pragma	Complete	✓ Low / No Risk
Timestamp Dependency for Crucial Functions	Complete	✓ Low / No Risk
Exposed _Transfer Function	Complete	✓ Low / No Risk
Transaction-Ordering Dependency	Complete	✓ Low / No Risk
Unchecked Call Return Variable	Complete	✓ Low / No Risk
Use of Deprecated Functions	Complete	✓ Low / No Risk
Unprotected SELFDESTRUCT Instruction	Complete	✓ Low / No Risk
State Variable Default Visibility	Complete	✓ Low / No Risk

The contract code is **verified** on BSCScan.

The vulnerabilities listed above were not found in the token's Smart Contract.

# Disclaimer



The opinions expressed in this document are for general informational purposes only and are **not intended to provide specific advice or recommendations for any individual or on any specific investment**. It is only intended to provide education and public knowledge regarding projects. This audit is only applied to the type of auditing specified in this report and the scope of given in the results. Other unknown security vulnerabilities are beyond responsibility. Dessert Finance only issues this report based on the attacks or vulnerabilities that already existed or occurred before the issuance of this report. For the emergence of new attacks or vulnerabilities that exist or occur in the future, Dessert Finance lacks the capability to judge its possible impact on the security status of smart contracts, thus taking no responsibility for them. The smart contract analysis and other contents of this report are based solely on the documents and materials that the contract provider has provided to Dessert Finance or was publicly available before the issuance of this report (issuance of report recorded via block number on cover page), if the documents and materials provided by the contract provider are missing, tampered, deleted, concealed or reflected in a situation that is inconsistent with the actual situation, or if the documents and materials provided are changed after the issuance of this report, Dessert Finance assumes no responsibility for the resulting loss or adverse effects. Due to the technical limitations of any organization, this report conducted by Dessert Finance still has the possibility that the entire risk cannot be completely detected. Dessert Finance disclaims any liability for the resulting losses.

Dessert Finance provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Even projects with a low risk score have been known to pull liquidity, sell all team tokens, or exit-scam. Please exercise caution when dealing with any cryptocurrency related platforms.

The final interpretation of this statement belongs to Dessert Finance.

Dessert Finance highly advises against using cryptocurrencies as speculative investments and they should be used solely for the utility they aim to provide.



# Thank You

DESSERT FINANCE LIGHT AUDIT HAS BEEN COMPLETED FOR DESSERTSWAP (DSRT)  
THIS AUDIT IS ONLY VALID IF VIEWED ON [HTTPS://WWW.DSSERTSWAP.FINANCE](https://www.dessertswap.finance)

[www.dessertswap.finance](http://www.dessertswap.finance)  
<https://t.me/dessertswap>